"Express Mail" mailing label number:

EL684226501US

METHOD AND APPARATUS FOR SPECULATIVE ARBITRATION

Hans Eberle
Nils Gura
Nicolas Fugier
Bernard Tourancheau

CROSS-REFERENCE TO RELATED APPLICATION(S)

BACKGROUND

Field of the Invention

[1001] This invention relates to sharing of resources and more particularly to more efficient allocation of resources.

Description of the Related Art

[1002] Arbitration is needed if a shared resource is simultaneously requested by several users of that resource and access to the resource can only be granted to one user at a time. Fig. 1 shows a 2 X 2 crossbar switch coupling two input ports IP1 and IP2 with two output ports OP1 and OP2. In the example of Fig. 1, arbitration is required to allocate the output ports OP1 and OP2 to the input ports IP1 and IP2 since the input ports can simultaneously request the same output port and an output port can be utilized for a data transfer only by one input port at a time.

[1003] Fig. 2 shows exemplary timing of an arbitrated access to a shared resource such as may occur when IP1 and IP2 both request use of the same output port for the same time period. As is typical in arbitrated system, access to the resource is granted in fixed discrete time intervals 201 and 203. Arbitration (ARB1 and ARB2) typically also occurs in fixed discrete intervals 205 and 207. Fixed discrete intervals are typically used for arbitration and access intervals to make coordination with other system resources simpler. Note that arbitration overlaps access in the example shown in Fig. 2. For example, arbitration period 207 (ARB2) for access interval 203 (ACC2)

overlaps access period 201 (ACC1). In the simple example of Fig. 2, there are two users issuing requests REQA and REQB for one shared resource. Requests REQA and REQB are considered during arbitration period ARB1 and request REQA is granted use of the shared resource during access period ACC1. REQB is again considered during arbitration period ARB2 and REQB is granted access to the shared resource during access period ACC2.

[1004] Unfortunately, there is unwanted delay between the time a request is generated (e.g. at 209) and the time the actual access begins. Assuming requests are generated at arbitrary times during an access interval and further assuming that there are no conflicts for the requested resource, the average delay is half the access interval plus the arbitration time. In the worst case, the delay can be the full interval plus the arbitration time.

[1005] Referring to Fig. 3, the worst case delay, given the assumptions above, is shown for request REQB, which misses arbitration cycle ARB2 and as a result has to wait to be considered in arbitration cycle ARB3. Note also that as a result, access interval ACC2 is not granted to anyone and therefore the resource goes unutilized during that access interval. Request REQB is subsequently granted the resource for use during access interval ACC3. The delay encountered by request REQB is unwanted since the delay increases the latency of the data, causes underutilization of the resource and can reduce throughput.

[1006] It would be desirable to provide an arbiter that more efficiently shares the available resources among the various users that request their use.

SUMMARY

[1007] Accordingly, the invention provides a method for sharing resources among multiple users of those resources using an arbiter. The arbiter speculatively allocates one or more of the shared resources to allow more efficient utilization of those resources. In one embodiment, the arbiter allocates one of the resources speculatively to one of the users for use during a particular access interval in the absence of a request for the resource from the user that is speculatively allocated the resource. The

method may further include allocating at least a second of the resources for use during the particular access interval according to a request received by the arbiter for the other resource. That is, a particular access interval may include both speculative and non-speculative allocations. The arbiter may speculatively allocate multiple resources to respective users or may speculatively allocate multiple resources to a single user.

[1008] Any of a variety of approaches can be used to determine which of the users is speculatively allocated a resource. For instance, the arbiter may speculatively allocate a resource to a user because the user was granted a request for use of the resource during a previous arbitration cycle. Alternatively, the arbiter may speculatively allocate a resource because the user requested the resource during a previous arbitration cycle.

[1009] In another embodiment the invention provides an apparatus that includes a plurality of resources and a plurality of users of those resources. The users contend for use of the resources. Arbitration logic is coupled to receive requests from the users for use of the resources. The arbitration logic speculatively allocates a resource to one of the users for use during an access interval, absent a request from the one user for the at least one resource. The arbitration logic may further allocate at least another of the resources for use during the access interval according to a request received by the arbitration logic for the other resource. The arbiter can utilize a variety of approaches to determine which of the users is speculatively allocated which resource. For example, the arbitration logic may speculatively allocate the at least one resource when the user to whom the resource is being speculatively granted has been granted a request for use of the resource during a previous arbitration cycle.

BRIEF DESCRIPTION OF THE DRAWINGS

[1010] The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

[1011] Fig.1 shows a 2X2 switch.

[1012] Fig. 2 shows an example of a typical arbitration sequence.

[1013] Fig. 3 shows an arbitration sequence in which access was delayed because the request came after the start of an arbitration cycle.

[1014] Fig. 4 shows an exemplary arbitration cycle in which a resource is speculatively granted.

[1015] Fig. 5 shows an exemplary crossbar switch which can exploit one or more embodiments of the present invention.

[1016] Fig. 6 shows a multi-processor system which can exploit one or more embodiments of the present invention.

[1017] Fig. 7 shows an exemplary schedule generated that includes a speculatively allocated resource.

[1018] Fig. 8 shows a schedule generated that includes a speculatively allocated resource using a hint.

[1019] Fig. 9 shows an example of a speculatively allocated real time request.

[1020] Fig. 10 shows an example of multiple resources allocated to a single user thereby allowing for speculatively allocated multicast operations.

[1021] Fig. 11 shows an alternative in which a fast arbiter is used in conjunction with a regular arbiter to handle late arriving requests for resources.

[1022] The use of the same reference symbols in different drawings indicates similar or identical items.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

[1023] Referring to Fig. 4, an example shows how an arbiter arbitrates requests conventionally and additionally speculatively assigns resources to users of the resources in the absence of requests for the resources. Since there is a delay between the generation of a request and the actual usage of the resource, speculative allocation

of a resource to a user whose request occurs after the beginning of the arbitration cycle (or even after the beginning of the access interval as explained further herein) increases performance in respect to latency and throughput if requests that occur during the arbitration cycle can be processed in the immediately following usage (or access) interval.

[1024] Fig. 4 shows how speculative allocation of resources can improve throughput and reduce latency. In arbitration cycle ARB1, request REQA(1) is granted the usage interval ACC1. User A, who requested the resource in REQA(1), uses the granted resource during usage interval ACC1. No requests have been received by the arbiter by the start of arbitration cycle ARB2. However, even though no requests have been received, the arbiter grants usage interval ACC2 speculatively to user A. Another request from user A (REQA(2)) occurs after the start of arbitration cycle ARB2. Because the usage interval ACC2 has been speculatively granted to user A, the second request from user A, REQA(2), can be serviced during usage interval ACC2, rather than having to wait for the arbitration cycle ARB3, thereby allowing utilization of the resource one cycle earlier than a non-speculative arbiter approach would provide. In one embodiment, each arbitration cycle includes a "regular" portion in which resources are allocated in response to requests for those resources, and a "speculative" portion in which those resources unallocated in the regular portion may be speculatively allocated according to any of a number of approaches described further herein.

[1025] Referring to Fig. 5, an embodiment of the invention is illustrated in which arbiter 501 schedules usage of shared resources, i.e., output ports 503, 505 and 507 among requesters, i.e. nodes 121, 123, and 125 connected to input ports, 509, 511 and 513, both in response to explicit requests and speculatively, that is, in the absence of a specific request from the user granted the resource. Crossbar switch 515 forwards packets from its input ports, 509, 511 and 513 to its output ports 503, 505 and 507. Each node can hold multiple packets destined for different output ports. In a typical arbiter, the switch schedule calculated connects input and output ports in such a way that as many packets as possible can be forwarded simultaneously, thus maximizing usage of shared resources. Each node sends a set of requests to arbiter 501, which

then chooses the requests to be granted such that resources are allocated to requesters in a conflict-free way.

[1026] When no requests are received for a particular resource, i.e. an output port, during a particular usage interval in the switch, arbiter 501 speculatively grants the output port to one of the requesters, i.e. one of the nodes for the usage interval. In one embodiment of the invention, the arbiter operates in a synchronous manner in that the arbiter receives request signals 517 for shared resources at the same time from the various nodes 121, 123 and 125. Scheduling happens synchronously in that grant signals 519 are sent at the same time and the usage interval for each resource has the same length. Scheduling may be further constrained in that only one requester can use a particular resource at the same time. When developing an arbitration scheme the main goal typically is to achieve high aggregate usage of the resources while still providing a minimum level of fairness, mainly in the sense that starvation of individual requests is prevented.

[1027] In the example shown in Fig. 5, node 121 has a request 122 for the use of output port 503, and node 123 has a request 124 for output ports 503 and 507. Node 125 does not have any requests. A variety of approaches may be utilized for the arbitration scheme to allocate the output ports to the input ports. Examples of such arbitration schemes include fixed priority schemes, round robin schemes, and prioritizing requesters based on the number of their requests. Assume that under some arbitration scheme output port 503 is allocated to input port 509 (node 121) and output port 507 is allocated to input port 511 (node 123). Output port 505 is thus available to be speculatively allocated to input port 513 (node 125).

[1028] Another example where a speculative arbitration scheme may be utilized is shown in Fig. 6, which is a multi-bus interconnection structure 600 functioning as a transport mechanism for connecting multiple processors 601, 603 and 605 (also referred to in Fig. 6 as P0, P1 and P2) with multiple memories 607, 609 and 611 (also referred to in Fig. 6 as M0, M1 and M2). Each processor can have multiple outstanding transactions such as read and write operations. Similar to the switch embodiment described in relation to Fig. 5, the bus schedule to be calculated connects

processors and memories in a conflict-free way, ideally, such that at a given time as many transactions as possible can be executed in parallel.

[1029] The arbiter 602 may be coupled to each of the buses shown in Fig. 6. The request and grant lines may be incorporated into the buses as well or may be separately coupled to the arbiter 602. In fact, the arbiter, rather than being separate as shown, may in fact reside in one or more of the processors or memories, or be distributed among the processors and memories.

[1030] Assuming a central arbitration scheme utilizing arbiter 602, processor P0 requests transaction T0 for memory M0 from the arbiter, processor P1 requests transactions T0 and T2 for memories M0 and M2, respectively. Processor P2 has no current requests. Assume that the arbiter grants P0 usage of memory M0, and P1 usage of memory M2 for a particular usage interval. Since P2 has not been granted a resource and M1 is not being utilized the arbiter may speculatively grant the usage of M1 to P2.

[1031] While several examples have been given for systems which may exploit the speculative arbitration described herein, arbitration is used in many electronic and communication systems, and the speculative arbitration scheme described herein is generally applicable to those systems in which arbitration for resources by users is required.

[1032] While it has been assumed so far that a user accesses a resource from the beginning to the end of the usage interval, partial use of a usage interval by a user is also possible. For example, an input port of a switch in Fig. 5 may transfer data to a granted output port only during a portion of the interval. If a system can provide capability to partially use a resource during an access interval, that can be exploited by allowing a user request that arises after the start of an access interval to be serviced for a portion of that interval when the resource has been speculatively granted to the user with the late request. Thus more efficient use of resources can be provided if partial utilization capability is combined with speculative allocation.

[1033] Fig. 7 shows an example of schedules calculated by an arbiter using speculative arbitration. During arbitration cycle 1, the requester REQ1 is granted a request for resource RES1. Requester REQ2 is not granted any resources and requester REQ3 is granted resource RES3. During the next arbitration cycle 2 requester REQ1 is granted resource RES1 and requester REQ3 is speculatively allocated resource RES3 without having requested. In the example in Fig. 7, the arbiter speculatively allocates RES3 to REQ3 because of the grant of RES3 to REQ3 in the previous arbitration cycle, where REQ3 indicates a user that did not make a request for the resource.

[1034] There are many different ways that can be utilized to determine how to speculatively allocate resources to users. For example, a speculative grant may be based on a previous grant or previous request for the resource in the immediately preceding (or earlier) arbitration cycle. While in some embodiments, only some of the ungranted resources may be speculatively allocated based on previous requests or grants, in other embodiments an arbiter may allocate as many unused resources as possible.

[1035] In some circumstances, multiple users may have requested a resource during one or more previous arbitration cycles. Many different approaches can be utilized to determine which of the users should get the speculative grant. For example, referring again to Fig. 4, one simple approach is to grant access to the user who requested the resource in the immediately previous arbitration cycle or to the last user to utilize the resource if the resource was not utilized in the previous interval. Thus, in Fig. 4, since user A requested the resource in ARB1, user A is speculatively granted the resource in ARB2 even through no request has been received.

[1036] Other approaches are possible to select among multiple users. For example, users may be chosen utilizing a round-robin approach, or may be selected randomly (pseudo-randomly as usually implemented). Users may be selected according to a fixed priority scheme, or may be selected using an arbitration scheme in which the user with the fewest requests is granted the resource first. The arbiter may chose to allocate a resource speculatively to the user that has had the most

requests for the resource during a predetermined number of previous arbitration cycles. The number of arbitration cycles utilized depends on the implementation of the arbiter and the application in which the arbiter is being used, and may vary from, e.g., two cycles, to tens or hundreds of arbitration cycles. In another embodiment, the arbiter may chose to allocate a resource speculatively to the user that has had the most grants for the resource during a predetermined number of previous arbitration cycles. The arbiter may chose to allocate a resource speculatively to the user that has had the most requests or most grants for all resources combined during a predetermined number of previous arbitration cycles.

[1037] The arbiter may chose to allocate a resource speculatively to the user that has had the highest fill-level for the send queue associated with the resource to be scheduled, the fill-level either being the current fill-level or the fill-level averaged over the last n arbitration cycles, where n is implementation and application dependent. For example, referring to Fig. 5, each node 121, 123 and 125 may have send queues associated with each of the output ports. The fill level for the particular queue can be sent to the arbiter 501 along with the request for each of the output ports.

[1038] The arbiter may choose to allocate a resource speculatively based on the fill-level of the receive queue associated with the resource to be scheduled, averaged over the last n arbitration cycles, where n is implementation and application dependent. For example, referring to Fig. 5, each node may have receive queues associated with one or more of the output ports. The arbiter may choose to allocate a resource if the fill level of the receive queue is high, which would imply that the resource has received a lot of data transfers and is likely to receive more. The fill level for the particular queue can be communicated to the arbiter 501 along with the request for each of the output ports, if each node 121, 123 and 125 is coupled both to an input port and an output port. If the nodes are not coupled to the output port, additional communication links, such as additional wires, can be provided to communicate fill levels. The particular choice of how to allocate resources based on fill level may be based on such factors as predicted resource utilization and other system characteristics.

[1039] The various approaches described above may be used to choose among all users or just among those users having requests during the previous one or more arbitration cycles, or just among those users having requests during the previous one or more arbitration cycles for the particular resource being speculatively allocated.

[1040] Many other approaches and combinations of the above described approaches can be utilized to determine how to allocate a resource to a user speculatively. Providing speculative allocation allows the resources such as output ports or memories to be more efficiently utilized by the users (e.g., nodes coupled to input ports and processors, respectively).

[1041] The arbiter may receive a hint from the users saying whether they want to be speculatively granted a resource or not. Any of the above described arbitration schemes or combination thereof, may be utilized to select among the users indicating in their hints that they wish to be allocated resources (or a particular resource) speculatively. Referring again to Fig. 5, the hints may accompany the requests sent by the nodes to the arbiter 501. The hints may indicate for each arbitration cycle the resource(s) in which the user is interested. Alternatively, the nodes may indicate to the arbiter in e.g., during an initialization procedure, whether they wish to be considered for speculative allocation of resources generally or for specific resources. In addition to supplying the hint, the resources may also specify a set of resources they would be interested in being speculatively granted. The arbiter may also receive a hint from a resource as to whether it wished to be speculatively granted. Referring to Fig. 8, an example is given of a schedule generated by an arbiter using hints. During the first part of an arbitration cycle requester REQ1 is granted resource RES1 and requester REQ3 is granted resource RES3. During a second part of the arbitration cycle, requester REQ2, who had provided a hint indicating that it wants to be speculatively granted resource RES2, is speculatively granted resource RES2.

[1042] An arbiter may utilize speculative arbitration to more efficiently allocate unused resources to periodic real-time requests such as video data, which require servicing within a fixed period of time. Referring to Fig. 9, the real time requirements are specified by period tp and jitter tj. tj defines the size of the window during which

any slot (usage interval) can be used to grant the real-time request. The user with the real time request may not notify the arbiter of the real-time request, speculating that the user will receive a speculative grant to fulfill the real-time request earlier than the deadline given by tp and tj. If there is not an unused slot, the last slot of the window will be allocated to the real-time request, thereby guaranteeing that the real-time request is serviced within its required period tp and jitter tj. In the example shown in Fig. 9, a request is granted in slot 1 and slot 12. The request is granted in slot 1 speculatively when the resource first becomes available in the jitter period. In slot 12 in contrast, the request is granted to meet the deadline given by tp and tj. This assumes that a real-time request does not conflict with any other real-time request and that real-time requests are given priority over non-real-time requests.

[1043] Note that while certain of the embodiments described herein may utilize fixed slots, the various embodiments described herein may also be used where variable sized slots are used.

[1044] A conventional arbiter typically coordinates grants such that a requester receives at most one grant. A speculative arbiter might decide to generate more than one grant per requester. That can be useful when applied to, e.g., network switches where multiple grants given to a single user correspond to granting a multicast connection. It is also useful when applied to unicast connections if more than one output port is available to be speculatively scheduled and a subset of output ports contains likely candidates of output ports to be speculatively allocated to the same input port. To make it possible for the user to only use a subset of the connections of a speculatively set up multi-cast, the transferred data items may have destination identifications attached or incorporated. The destinations can use that information to filter the received data appropriately. Referring to Fig. 10, input port IP1 is granted a multicast connection to output ports OP1, OP3, and OP4. The transferred data includes information identifying the destination as output ports OP1 and OP3. Thus, output port OP4 filters out the received data on that basis. In that way a speculative multicast may be set up and the user can use a subset of the speculatively allocated resources. Note that the arbiter may be implemented in software, hardware or a

combination of both. The particular implementation may vary according to the application in which the arbiter is being used and according to the algorithm(s) utilized by the arbiter in speculatively granting (and otherwise granting) resources.

[1045] Referring to Fig. 11, an alternative that may be used in place of or in addition to speculative arbitration is illustrated. The arbiter described in Fig. 11 considers "late requests." More specifically, once regular arbitration has taken place, late requests, that is requests received after the regular arbitration has started are considered by a "fast arbiter." The fast arbiter uses a less efficient (e.g., in terms of the number of granted requests) but faster algorithm than the regular arbiter. As shown in Fig. 11, while the request REQA(1) is considered by the regular arbiter during arbitration ARB1, REQA(2), which is received after the start of the regular arbitration cycle ARB2, is considered by the fast arbiter in the fast arbitration portion FARB2, of the arbitration cycle.

[1046] The description of the invention set forth herein is illustrative, and is not intended to limit the scope of the invention as set forth in the following claims. The various embodiments and arbitration approaches described above may be used in various combinations. While specific examples of switches and multiprocessor systems have been shown, the arbitration approach described herein is generally applicable to sharing of resources in a variety of systems including networks and communication systems where resources are shared. Other variations and modifications of the embodiments disclosed herein, may be made based on the description set forth herein, without departing from the scope of the invention as set forth in the following claims.